14th European Signal Processing Conference (EUSIPCO 2006), Florence, Italy, September 4-8, 2006, copyright by EURASIP

FAST INTEGER ARITHMETIC WAVELET TRANSFORM PROPERTIES AND APPLICATION IN FPGA/DSP SYSTEM

*Wojciech Półchłopek, Wojciech Maj and Wojciech Padee**

Department of Electronics, AGH University of Science and Technology; *Warsaw University of Technology
al. Mickiewicza 30, 30-059 Cracow, Poland; *ul. Nowowiejska 15/19, 00-665 Warsaw, Poland
phone: + (48) 12 617 27 00, fax: + (48) 12 617 30 45, email: ph@agh.edu.pl

ABSTRACT

The new fully integer processing of the wavelet scheme compression enables very fast application and thus it can be very useful for application in real-time systems. The most important property of this concept seems to be the possibility of simple and fast application into FPGA chip. The new Fast Integer Arithmetic Wavelet Transform (FIAWT) can be a very useful tool to compute DWT (and non-decimated DWT) for the time-restricted systems (real-time data processing) e.g. Data Acquisition (DAQ) systems with a waveform recorder. In this paper the authors show some important aspects of FIAWT. The application example (FPGA/DSP) in the ICARUS [1] DAQ system for compression with signal recognition is included as part of this paper.

1. INTRODUCTION

Standard application of Wavelet Transform which maps integers to integers, uses floating-point arithmetic with smart rounding to compute the transform [1,7]. This often means redundant processing which requires very complicated processing architecture and so it is very time-consuming. Most of the processing stages can be omitted or simplified using an adaptation to the integer arithmetic processing. In most cases it is possible to apply the transform using only integer addition and bit shifting operations. This can result in ultra-fast processing especially when implemented into FPGA (the bit shift arithmetic does not require any additional logic). The application of this FIAWT is currently in phase of testing in the ICARUS DAQ system and shows very good compression quality and processing efficiency in the scattered multi processing unit environment. The next step to increase the overall efficiency is foreseen as an implementation of this scheme to the multi channel signal with time division and wavelet-domain time windowing ("zero skipping" algorithm) of oversampled DWT which is currently in the phase of designing and testing.

2. WAVELET TRANSFORM WHICH MAPS INTEGERS TO INTEGERS

The lifting scheme [7] is considered to be an efficient implementation of filtering operations at each level when computing a discrete wavelet transform. This is true when compared with a standard DWT application. Anyway it can be further simplified and sped-up in case of the integer version of the transform.

The standard wavelet transform which maps integers to integers [1,4] uses floating-point processing with smart rounding on every lifting stage. The equations below 1, 2 and 3 show the integer lifting algorithm while figure 1 shows the application scheme.

Splitting: $$s_j : s_{j(e)}, s_{j(o)} \quad (1)$$

Prediction: $$d_{j-1} = s_{j(o)} - floor(P\{ s_{j(e)}\}+0.5) \quad (2)$$

Update: $$s_{j-1} = s_{j(e)} + floor(U\{ d_{j-1}\}+0.5) \quad (3)$$

where: floor means rounding down to integer (omitting the fractional part of data)

This standard algorithm can be computationally inefficient and often requires a very fast floating point processing unit to apply in the real-time. The floating-point operations are very "hardware-consuming" and considerably slow when applied in FPGA or VLSI chip.

The whole algorithm can be implemented using only integer arithmetic - every floating-point division can be replaced by bit shifting and the rounding stage is omitted. In certain cases of DWT (i.e. linear prediction), the whole algorithm can be applied using only a few integer processing operations and thus ultra-fast processing can be achieved. Of course in this case a nonlinear transform is obtained, but in any case integer version of WT is nonlinear [1, 4]. Anyway, it is possible to find a transform which can be even more complicated when applied in FIAWT, in comparison with standard floating-point arithmetic.

This approach can be the fastest application of the DWT and can be also easily implemented in VLSI or FPGA chip. Complex floating-point multiplication and smart rounding to integers can be replaced by simple and fast binary shifting and integer adding.

3. FAST INTEGER ARITHMETIC WAVELET TRANSFORM

It is intuitively clear that every rational number from 0 to 1 can be well approximated by weighing sum of the negative

powers of two, e. g. this is the concept of fractional arithmetic of DSP processors:

$$c_m = \frac{k}{l} \approx \sum_{n=1}^{M} a_n \left(\frac{1}{2}\right)^n, \quad k,l \in N^+, k \le l \qquad (4)$$

where: M - number of bits for approximation (maximum order of the shifting stage in the FIAWT algorithm), $a_n \in \{0,1\}$;

According to equation 4 , every floating-point multiplication by DWT filter coefficient (from 0 to 1) with rounding can be computed as a sum of binary shifted data (see fig. 3 and 4). Depending on the coefficient value or approximation quality a transform more or less complicated in application can be obtained. One of the easiest to apply is Linear Prediction Interpolating Wavelet Transform - biorthogonal (2,2) - see fig. 4.

This transform has been applied to the ICARUS DAQ system and all the compression results were obtained with this ultra fast and simple transform [8]. The concept showed in fig. 3 is easy to simplify (figure 4 shows how the specific transform can be simplified). The whole transform (2,2) can be computed using only four integer adding blocks and two bit shifters. In the same way most of the transforms can be simplified, although it is possible to obtain a scheme more complicated in application, especially for the higher order transform which uses high precision coefficients (expressed by large number of bits).

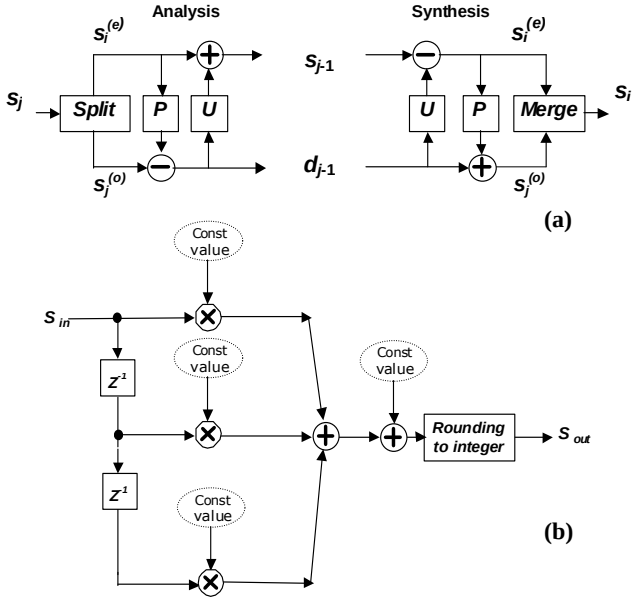

Figure 1 - Lifting scheme application (a) and implementation of the standard prediction and update filters for the integer transform (b)

According to the equations 2 and 3, the smart rounding to integers should be done by adding the constant value equal 0.5 before omitting the fractional part of the number. This approach enables rounding in the correct way.

This is also important when bit shifting is used – this shifting is also some kind of rounding so the correct way is to shift right data which is "smooth shifted left" by adding the small constant value. This "constant value" depends on the shifting order – and equals:

$$Const_{value} = 0.5 \cdot 2^N \qquad (5)$$

where: N – order of the shifting stage

It is easy to see that it is possible to use the same shifters in several "SMART Bit shift & add" blocks as well as constant value adders – see fig. 4. This approach can simplify the whole algorithm and its application, and thus reduce "hardware-consumption" and increase overall speed.

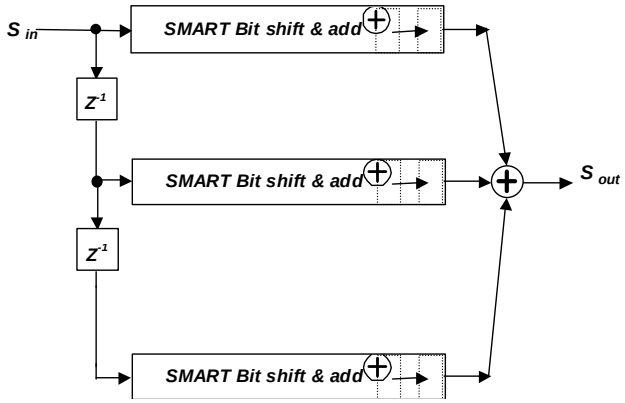

Figure 2 - FIAWT implementation of the prediction and update filters for the integer transform

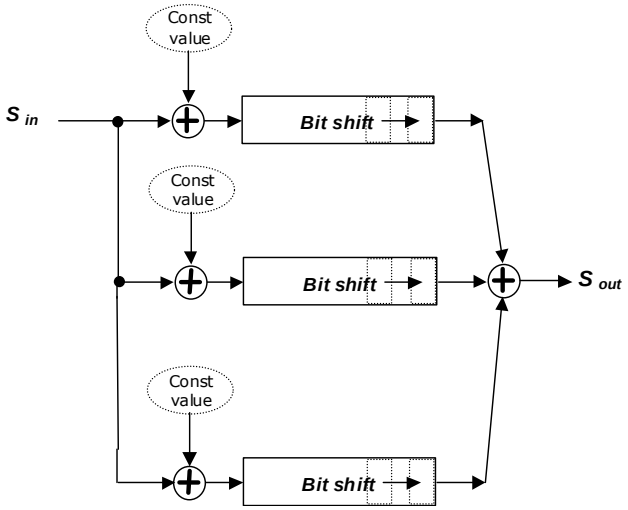

Figure 3 - SMART Bit shift & add block - application scheme

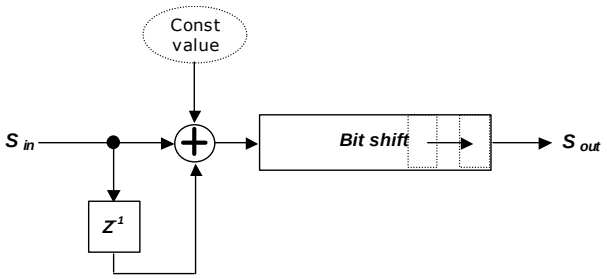

Figure 4 - Predictor (shift by one bit, constant value equal to 1) and update block (shift by two bits, constant value equal to 2) for the linear prediction biorthogonal (2,2) and reverse bior(2,2)

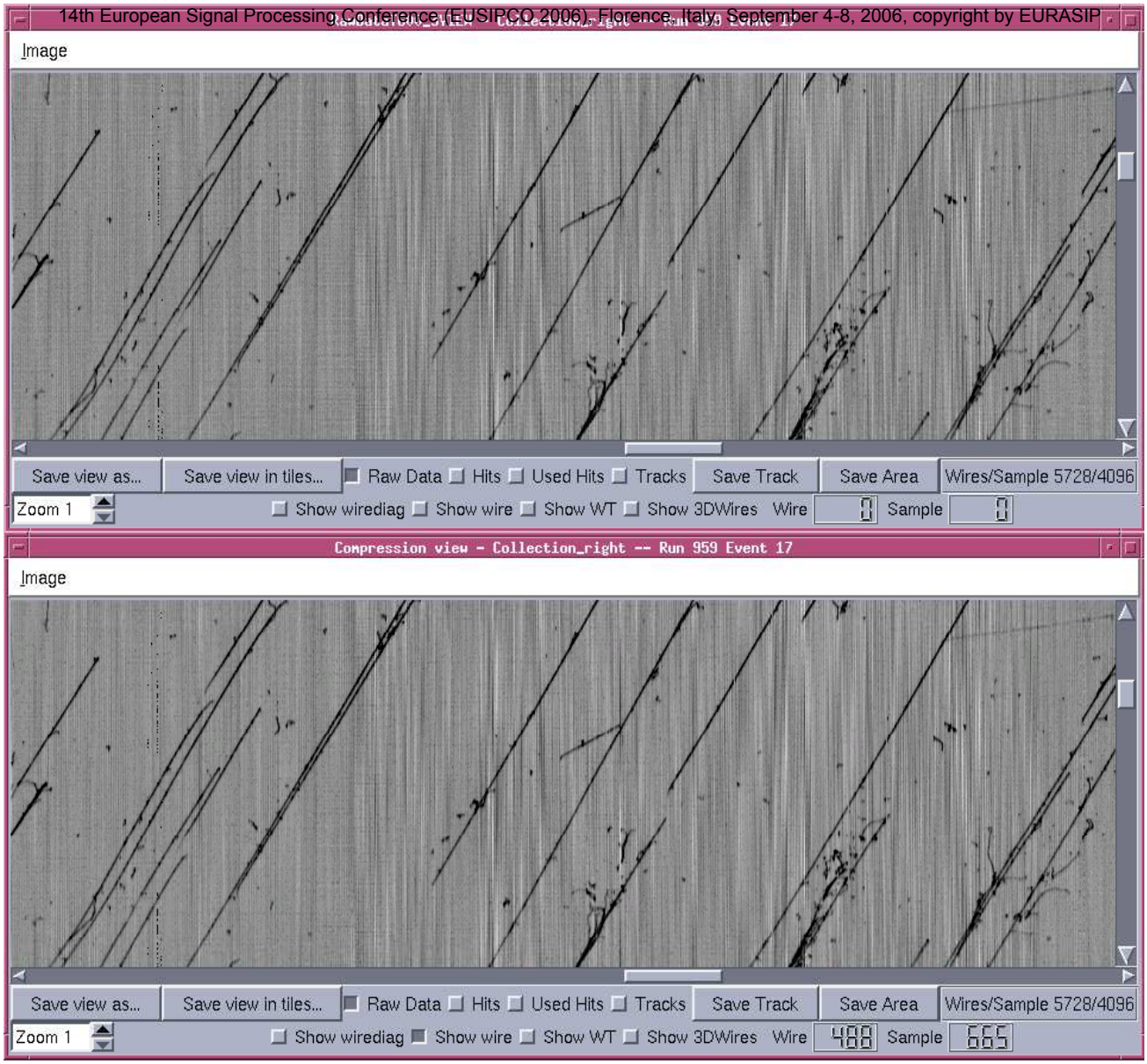

Figure 5 - 2-D compression views of the ICARUS event file [2] (220MB compressed to 3.5MB) – compression for this plane 42.50 times. Upper - original file, lower - processed file.

4. COMPRESSION RESULTS

The initial state simulations were done in the Matlab® environment. Compression results were obtained by the set of programs written in the Matlab environment and then compiled to C++ by the Matlab Compiler® and finally via C++ data analysis program called Qscan[3] with the compression interface. This application is designed for processing the ICARUS [9] [10] raw data files [2,3].

Figures 5 and 6 show "the subjective compression quality" (views of hard to compress event 959 obtained using the Qscan[3] program) which can be considered to be very high while the compression factor is also high (30-70). Two dimensional image-like views (fig. 5) show almost no difference, small differences between the original and compressed signal can be seen on the one dimensional view (see fig. 6). This "subjective quality" remains good also for the higher compression ratios while the real quality-degradation results during extraction of the physical parameters from the data – for details see figure 7. The most important parameter P3 corresponds to the Δ_{mp} – particle energy deposit and has been reconstructed with high accuracy (1,2% mean error for CR=30); parameters P2 (σ) and P4 (ξ) have been obtained as the Landau distribution fitting parameters and are of much lower importance [9].

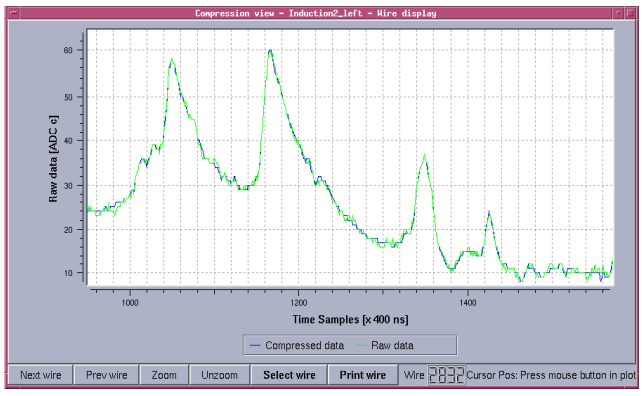

Figure 6 - 1-D compression view – current wire (1-D vertical scan) CR = 30.2

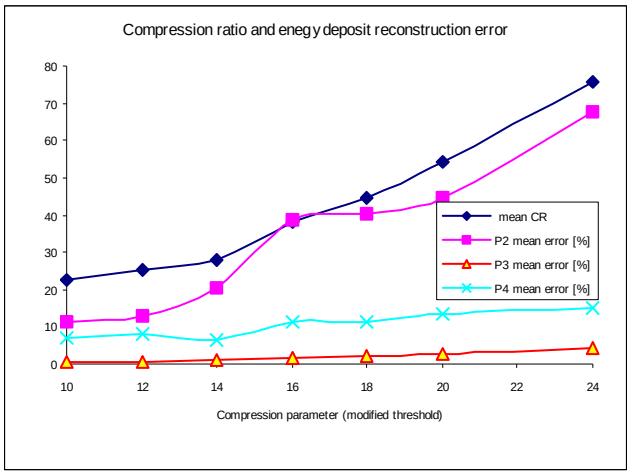

Figure 7 – Compression quality: Compression ratio and degradation of the Landau distribution fitting parameters (P3 – energy deposit of particle)

5. FIAWT FPGA IMPLEMENTATION

The real-time compression application is based on FIAWT (bior(2,2)) and thresholding technique with fast signal recognition based on oversampled FIAWT (rbio(2,2)) [8],[9] applied in the FPGA.

The FPGA DWT application is based on VHDL behaviour description of the DWT block shown in figure 1 (analysis block) and figure 4 (simple filter).
The module input and output signals are described below:

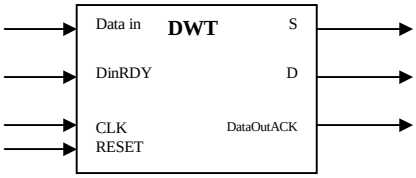

Figure 8 –Input and output signals description

The DWT module operates in a synchronous way, and fully implements digital signal analysis block shown in figure 1 with one data input, two data outputs (both for S and D signals), system clock and reset signals and additional two control signals (used for data synchronization).

Data input (as well as data asserted to S output) is 10 bit wide and is interpreted by the device as an unsigned number. Data output "D" (detail) is a 10 bit signed number. The control signals DinRDY and DataoutACK take part in data flow process: input data is accepted when DinRDY is HIGH on the rising edge of the system clock. DataOutACK function is to provide information to the other devices (next stage DWT) connected to this module when the results are ready.

The whole device is controlled by a simple Finite State Machine (FSM), whose main tasks are to gather data delivered to the device input, assert results on the outputs and read/assert control signals. Its functional diagram is shown above (fig. 9). The FSM waits until two data samples are delivered to the device input, then it activates arithmetic modules which are responsible for the algorithm implementation (FSM implements “mirror extension” by doubling the first input sample), then asserts calculation results and control signals to theirs outputs.

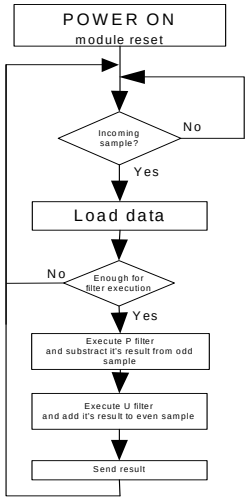

Figure 9 – Control module flow diagram

Before the device is ready to accept new data (next two samples) a three clock cycles delay is required (for arithmetic modules and FSM). This three clock cycles delay condition was met by assuming that the device clock is at least three times faster than the input data clock.

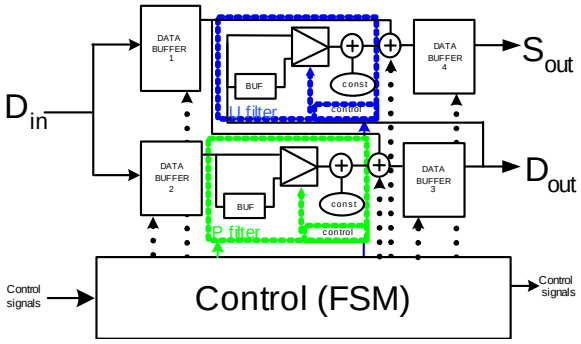

Figure 10 – Simplified post-synthesis device schematic

The simplified post-synthesis schematic is shown in figure 10. The P and U filter, data buffers and main FSM are clearly visible as separate modules. Every single action, such as latching data into input buffers, activating P and U filters, adders and intermediary buffers, is initiated by control signals delivered from FSM module. The filter schematic can be compared with its functional description shown in figure 4 to see how each processing step was implemented: the delay module shown in figure 4 was implemented as a single buffer with a multiplexer box, and data shifting as a simple bus rerouting with a sign extension.

The synthesized project was implemented into the specific FPGA device - Xilinx XC3S1500FG456-4 (speed grade 4 - 3,328CLBs). The final FPGA design consists of eight stage DWT (bior (2,2)) and six stage simplified oversampled DWT (rbio (2,2)) used for data recognition in each of the sixteen data channels sampled at 2,5MHz (multiplexed into 40 MHz data stream) [10]. Each DWT stage resulted in the utilization of 54 slices (4 slices = 1 CLB) and could operate at frequencies up to 50MHz. The test project (which consists of simple DWT stage module and additional test module) was able to operate at frequencies up to 75 MHz

6. CONCLUSIONS

The new fully integer processing of the wavelet scheme compression enables a very fast application and thus it can be very useful in the application in real-time systems. The most important property of this concept is the possibility of a simple and fast application into FPGA or ASIC chip. Until now the online compression has been applied in part of the DAQ system only for testing purposes. The final application is foreseen in scattered multiprocessing units DAQ architecture in mixed TI-DSP/Xilinx-FPGA system.

7. ACKNOWLEDGMENTS

The work reported in this paper is supported by the Polish national grant: 3 T11C 008 27. Many thanks for the ICARUS Collaboration - especially to Sandro Centro (INFN Padova) and Agnieszka Zalewska (IFJ Krakow) for cooperation support.

REFERENCES

[1] Sweldens W., Schroder P., "Building your own wavelets at home", Wavelets in Computer Graphics, pages 15-87, ACM SIGGRAPH Course notes, 1996

[2] Uytterhoeven G., Roose D., Bultheel A., "Wavelet transforms Using the Lifting Scheme", Report ITA – Wavelets –WP1.1, 1997

[3] Donoho D. L., "Interpolating wavelet transforms" Preprint, Department of Statistics, Stanford University 1992

[4] Caldebank A. R., Dauberchies I., Sweldens W., Boon-Lock Yeo, "Wavelet transforms that map integers to integers", Technical report, Department of Mathematics, Princeton University 1996

[5] Sweldens W., "The lifting scheme: A custom-design construction of biorthogonal wavelets", Appl. Coput. Harmon. Anal., 3(2), pp. 186-200 1996

[6] D. L. Donoho, I. M. Johnstone, "Adapting to unknown smoothness via wavelet shrinkage", J. Amer. Statist. Assoc., 90, pp. 1200–1224, 1995

[7] Daubechies I., W. Sweldens, "Factoring wavelet transforms into lifting steps", J. Fourier Anal. Appl., 4 (3), pp. 245–267, 1998

[8] W. Półchłopek, M. Ziółko, "Wavelet Transform Compression and Denoising in Real-Time System" Proceedings of CNDSP Conferrence, Stafford, pp. 141-148, 2002

[9] Półchłopek W., Ventura S., Pietropaolo F., "Wavelet Transform Compression and Denoising in Real-Time System (Proposal for the ICARUS DAQ System)" ICARUS TM2002/12 , Padova 2002 - ICARUS collaboration internal note: for pdf copy write to author

[10] S. Amerio,..., W. Półchłopek ,.... (ICARUS Collaboration) "Design, construction and tests of the ICARUS T600 detector", Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, Volume: 527, Issue: 3, July 21, 2004, pp. 329-410.

[1] ICARUS (Imaging Cosmic And Rare Underground Signals) is one of the biggest experiments in the nuclear physics – for details see www pages at: http://www.aquila.infn.it/icarus/

[2] ICARUS packet raw data files consist mostly of the data packets of 16 channels 4096 samples 16 bits each. The whole event file is more than 220MB of data (over 13000 channels).

[3] Qscan is a C++ Qt based application written by ICARUS collaboration for offline data viewing, analysing and processing.

For ICARUS collaboration members see L'Aquila pages at: http://www.aquila.infn.it/icarus/